Swarm and Evolutionary Computation 14 (2014) 31–47

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Survey Paper

Population-based metaheuristics for continuous boundary-constrained dynamic multi-objective optimisation problems

Mardé Helbig [a,b,*], Andries P. Engelbrecht [b]

[a] Meraka Institute, CSIR, Scientia, Meiring Naudé Road, 0184 Brummeria, Pretoria, South Africa
[b] Department of Computer Science, University of Pretoria, Pretoria, South Africa

ARTICLE INFO

Article history:
Received 12 October 2012
Received in revised form
26 August 2013
Accepted 28 August 2013
Available online 27 September 2013

Keywords:
Population-based algorithms
Dynamic multi-objective optimization
Dynamic multi-objective optimization algorithms

ABSTRACT

Most real-world optimisation problems are dynamic in nature with more than one objective, where at least two of these objectives are in conflict with one another. This kind of problems is referred to as dynamic multi-objective optimisation problems (DMOOPs). Most research in multi-objective optimisation (MOO) have focussed on static MOO (SMOO) and dynamic single-objective optimisation. However, in recent years, algorithms were proposed to solve dynamic MOO (DMOO). This paper provides an overview of the algorithms that were proposed in the literature to solve DMOOPs. In addition, challenges, practical aspects and possible future research directions of DMOO are discussed.

1. Introduction

Most of the current research in the field of multi-objective optimisation (MOO) are focussed on optimisation problems where all of the sub-objectives are static [1–4]. On the other hand, research on solving dynamic optimisation problems is strongly focussed on dynamic single-objective optimisation problems (DSOOPs) [5–8].

However, optimisation problems that occur in situations of everyday life are normally not static in nature and have many objectives that have to be optimised, i.e. dynamic multi-objective optimisation problems (DMOOPs). One example of a real-life DMOOP is a steel production plant, where customers place an order for specific products that have to be delivered by a specified date. In order to produce a customer's order, the material has to go through specific production lines. Each production line consists of a number of machines that can only manage a certain load. Since many orders' materials are managed in the production lines at the same time, and some orders may require the same machines, the order in which the material of the various orders moves through the production line has to be optimised. Since machines can break down requiring the production lines to be re-optimised, the optimisation of a production plant is an example of a DMOOP. Other examples include route optimisation according to real-time traffic, resource management of a hospital (e.g. assigning patients to an operating room or the intensive care unit (ICU)) and the optimisation of indoor heating, i.e. regulating the indoor temperature as efficiently as possible.

Multi-objective optimisation problems (MOOPs) with conflicting objectives do not have a single solution. Therefore, multi-objective algorithms (MOAs) aim to obtain a diverse set of non-dominated solutions, i.e. solutions that balance the trade-off between the various objectives, referred to as the Pareto-optimal front (POF). Another goal of MOAs is to find a POF that is as close as possible to the true POF of the problem.

DMOOPs are MOOPs with at least one objective changing over time. Therefore, algorithms solving DMOOPs must have the ability to track the changing POF in order to find non-dominated solutions that are close to the new true POF.

This paper provides an overview of population-based algorithms developed to solve continuous boundary-constrained DMOOPs. The most important concepts of MOO and dynamic multi-objective optimisation (DMOO) are provided in Section 2. Section 3 provides an overview of dynamic MOAs (DMOAs) that have been proposed in the literature. Section 4 discusses challenges, practical aspects and possible future research directions of DMOO, as well as measuring the performance of DMOAs. Finally, a summary is provided in Section 5.

2. Definitions

This section provides definitions that are required as background for the rest of the paper. Sections 2.1 and 2.2 provide

* Corresponding author at: Meraka Institute, CSIR, P.O. Box 395, 0001 Pretoria, South Africa.
E-mail addresses: mhelbig@csir.co.za (M. Helbig), engel@cs.up.ac.za (A.P. Engelbrecht).

2210-6502/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2013.08.004

definitions with regards to MOO and DMOO respectively. The various DMOOPs and DMOO performance measures are not discussed in this section. However, the reader is referred to [9,10] for a comprehensive overview of the benchmark functions and performance measures proposed in the DMOO literature.

2.1. Multi-objective optimisation

The objectives of a MOOP are normally in conflict with one another, i.e. improvement in one objective leads to a worse solution for at least one other objective. In order to determine whether one decision vector is better than another decision vector, decision vector domination is used. For MOOPs, when one decision vector dominates another, the dominating decision vector is considered as a better decision vector.

Let the n_x-dimensional search space or *decision space* be represented by $S \subseteq \mathbb{R}^{n_x}$ and the feasible space represented by $F \subseteq S$, where $F = S$ for boundary-constrained optimisation problems. Let $\mathbf{x} = (x_1, x_2, \dots, x_{n_x}) \in S$ represent the *decision vector* (a vector of the decision variables), and let a single objective function be defined as $f_k : \mathbb{R}^{n_x} \to \mathbb{R}$. Then $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}), \dots, f_{n_k}(\mathbf{x})) \in O \subseteq \mathbb{R}^{n_k}$ represents an *objective vector* containing n_k objective function evaluations, and O is the *objective space*.

Using the notation above, decision vector domination is defined as follows:

Definition 1. *Decision vector domination*: Let f_k be an objective function. Then, a decision vector $\mathbf{x}_1$ dominates another decision vector $\mathbf{x}_2$, denoted by $\mathbf{x}_1 \prec \mathbf{x}_2$, if and only if

- $\mathbf{x}_1$ is at least as good as $\mathbf{x}_2$ for all the objectives, i.e. $f_k(\mathbf{x_1}) \le f_k(\mathbf{x_2})$, $\forall k = 1, \dots, n_k$; and
- $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ for at least one objective, i.e. $\exists i = 1, \dots, n_k : f_k(\mathbf{x_1}) < f_m(\mathbf{x_2})$.

The best decision vectors are called Pareto-optimal, defined as follows:

Definition 2. *Pareto-optimal*: A decision vector $\mathbf{x}^*$ is Pareto-optimal if there does not exist a decision vector $\mathbf{x} \neq \mathbf{x}^* \in F$ that dominates it, i.e. $\nexists m : f_k(\mathbf{x}) < f_k(\mathbf{x}^*)$. If $\mathbf{x}^*$ is Pareto-optimal, the objective vector, $\mathbf{f}(\mathbf{x}^*)$, is also Pareto-optimal.

The set of all the Pareto-optimal decision vectors is referred to as the Paretooptimal set (POS), defined as

Definition 3. *Pareto-optimal set*: The POS, P^*, is formed by the set of all Pareto-optimal decision vectors, i.e.

$$P^* = \{\mathbf{x}^* \in F \mid \nexists \mathbf{x} \in F : \mathbf{x} \prec \mathbf{x}^*\} \tag{1}$$

The POS contains the best trade-off solutions for the MOOP. The set of corresponding objective vectors are called the POF or Pareto front, which is defined as follows:

Definition 4. *Pareto-optimal Front*: For the objective vector $\mathbf{f}(\mathbf{x})$ and the POS P, the POF, $PF^* \subseteq O$ is defined as

$$PF^* = \{\mathbf{f} = (f_1(\mathbf{x}^*), f_2(\mathbf{x}^*), \dots, f_{n_k m}(\mathbf{x}^*)) \mid \mathbf{x}^* \in P\} \tag{2}$$

2.2. Dynamic multi-objective optimisation

Using the notation defined in Section 2.1, a boundary-constrained DMOOP is mathematically defined as

$$\begin{aligned} \text{minimise}: &\quad \mathbf{f}(\mathbf{x}, \mathbf{W}(t)) \\ \text{subject to}: &\quad \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \end{aligned} \tag{3}$$

where $\mathbf{W}(t)$ is a matrix of time-dependent control parameters of an objective function at time t, $\mathbf{W}(t) = (\mathbf{w}_1(t), \dots, \mathbf{w}_{n_m}(t))$, n_x is the number of decision variables, $\mathbf{x} = (x_1, \dots, x_{n_x}) \in \mathbb{R}^{n_x}$ and $\mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x}$ refers to the boundary constraints.

The goal of an algorithm solving a DMOOP is to track the POF over time, i.e. for each time step, to find

$$PF^*(t) = \{\mathbf{f}(t) = (f_1(\mathbf{x}^*, \mathbf{w}_1(t)), f_2(\mathbf{x}^*, \mathbf{w}_2(t)), \dots, f_{n_k}(\mathbf{x}^*, \mathbf{w}_{n_k}(t))) \mid \mathbf{x}^* \in P(t)\} \tag{4}$$

3. Dynamic multi- objective algorithms

This section discusses algorithms that have been proposed for DMOO. Algorithms that aggregate the objective functions of the DMOOP to create a DSOOP are not considered.

Section 3.1 discusses algorithms that solve DMOOPs without adapting the problem. Approaches that convert a DMOOP into multiple static MOOPs (SMOOPs) are discussed in Section 3.2. Section 3.3 discusses prediction-based approaches where knowledge of previous environments is used to predict the new POS or POF.

3.1. Algorithms that solve the dynamic multi-objective optimisation problem without adapting the problem

This section discusses static MOO (SMOO) algorithms that were adapted to solve DMOOPs and new population-based algorithms that were introduced for DMOO. Generic extensions that can be applied to any DMOA are also discussed.

3.1.1. Multi-objective optimisation algorithms adapted for dynamic multi-objective optimisation

This section discusses evolutionary algorithms (EAs) and particle swarm optimisation (PSO)-based algorithms that were proposed in the literature to solve SMOOPs and were adapted for DMOO.

Evolutionary algorithms: One of the first algorithms proposed to solve DMOOPs without using the weighted sum approach to aggregate the objective functions into a DSOOP was a hybridised minimal cost evolutionary deterministic algorithm (HMCEDA) introduced by Farina et al. [11]. This hybrid algorithm uses an $(1+1)$ evolution strategy (ES) for global optimization of the DMOOP [12]. An $(1+1)$ ES is an EA that applies Gaussian mutation [13] at each iteration to one parent to create one offspring, i.e. a random value from a Gaussian distribution is added to each element of a parent's vector to create an offspring. Once the $(1+1)$ ES starts to converge (determined by comparing the decision variable values of two consecutive iterations), a gradient-based algorithm or a simplex Nelder–Mead search algorithm [14] is used. HMCEDA was evaluated on the FDA DMOOPs [11]. For FDA1 and FDA2, the algorithm tracked the changing POF well over time and converged quickly to the new POF after a change in the environment occurred. However, for FDA3, HMCEDA struggled to converge towards the changing POF and struggled to find a diverse set of solutions. For FDA4, the algorithm converged reasonably well to the new POF after each change in the environment. However, for FDA5 where the density of the solutions changes over time, HMCEDA struggled to maintain a diverse set of solutions [11]. According to Farina et al. [11] the results from the study indicate that HMCEDA should use an EA with better performance, such as NSGA-II [15,16], for the global optimization.

Many SMOO algorithms were adapted for DMOO. Avdagić et al. [17] extended the multi-objective genetic algorithm (MOGA) to solve DMOOPs. MOGA, introduced by Fonseca and Flemming [18], was the first multi-objective genetic algorithm (GA) and incorporates Pareto-ranking. The advantages of MOGA are a simple fitness assignment